colorcue

a publication for Compucolor and Intecolor users • Dec., 1980/Jan., 1981 • $2.00

• The CALL function • Introduction to FORTRAN • Comp-u-writer • Compucolor Bell

About the cover . . . Colorcue celebrates its second anniversary with this issue, and thanks its many readers for their interest and support.

Colorcue is published every other month for users of the Compucolor II personal computer and users of Intecolor computers by Intelligent Systems Corp. Address all Colorcue correspondence to ISC, 225 Technology Park/Atlanta, Norcross, Georgia, 30092. Subscriptions to Colorcue are $12.00 per year in the U.S., Canada, and Mexico; and $24.00 elsewhere.

Contributing to the success of this issue:

Editor —
Susan Sheridan

Software & Hardware —
Gene Boughey
Knox Pannill
Myron Steffy
Heath Thompson
Bruce Williams

Art Direction —
Henry Wood

EDITOR'S LETTER

And now we are two! This DEC/JAN issue marks Colorcue's second anniversary and we're proud of the progress we've made. When Colorcue started in 1978, it was little more than a corner-stapled handout that was mailed to the few Compucolor II owners then in existence. Now we're larger, more informative, and more widely read than we ever imagined we would be.

This anniversary issue inaugurates a new Colorcue editor. Actually, it re-inaugurates a former editor — Susan Sheridan. Some of you will remember her from those earliest issues of Colorcue. It seems that Susan just couldn't stay away from Compucolor! She has returned to manage Marketing Communications for Intelligent Systems Corporation, and is very excited about being able to work on Colorcue again.

Much of the progress that Colorcue has made is due to the tremendous efforts of Cathy Abramson. She helped transform Colorcue into a smooth publication that really serves the users' needs. It was a big accomplishment and her efforts have been appreciated by users everywhere.

And now we are two! The editor's name is not the only change we've made in Colorcue. We've done a little re-vamping in content, form, and publication frequency as well. Now Colorcue will also be serving the Intecolor users. We'll be footnoting our COMPUCOLOR II programs with the changes necessary to make them run on Intecolor equipment. And we'll be accepting suggestions and contributions from our broadly-based group of Intecolor owners. We expect this to expand Colorcue's utility and help it reach even more users with common interests.

In order to make Colorcue more responsive to your needs, we have moved the Publications Department back in-house, where our staff will be right in the thick of things, working next to the people who have the answers. The improved information flow will help Colorcue get news to you more directly. In order to give us the time necessary for careful proofreading and absolute deadlines, Colorcue will be published every other month. (There were no AUG/SEP or OCT/NOV issues, and consequently everyone's subscription will be extended by two issues.)

We are firmly committed to supporting the Compucolor users through Colorcue. We think that the new changes we've made in the magazine will help us do just that. We always welcome your ideas and suggestions. Address all correspondence to: Compucolor Corporation, 225 Technology Park/Atlanta, Norcross, GA 30092, ATTN: Susan Sheridan.

It's great to be back!! *Susan*

REM

also wanted

Many of you have seen the full page 'WANTED' poster that Texas Instruments has been placing in the trade magazines this month and last. TI offers a reward for any software which they accept for marketing. Most of you know that we, too, offer a reward for software which we buy from users. If you have a program or two that you would like to submit for evaluation, send it to us, in Norcross, to the attention of Gene Boughey. If you wish, of course, we will gladly sign a non-disclosure agreement. Prices paid for software vary greatly, from one-time flat fees to royalty arrangements. If you have written a program that's useful to you, it may be useful to others. Send it in and let us have a look.

ADVANCED APPLICATION

assembly language — part 7

The assembly language programmer is frequently confronted with the problem of interfacing with the user. We have discussed such interface routines as CI, LO and OSTR in the past; however, these routines do not always satisfy all of the requirements. In this article, we will look at a routine to get a line of user input.

The GLINE routine allows the user to enter and edit the input line until the line is terminated by a CR (carriage return) or the input is aborted by a Control C. Upon termination, control is passed back to the calling routine with the status indicated by the 'Z' and 'C' flags. If neither the 'Z' nor the 'C' flag is set, then there is an input line of non-zero length in the input buffer and it is terminated with an end-of-line marker (00H). If the 'Z' flag is set, then the input line is of zero length and if the 'C' flag is set, a Control C was encountered and the input was aborted.

The values passed to the GLINE routine are the address of the input buffer, the address of the prompt message (containing an 'erase line' and ending with 239) and the length of the input buffer. The routine will allow the input of Length-1 characters. This is done so that there will be a place for the end-of-line marker upon exit. When an attempt is made to input any more than the allowed number of characters, they will be ignored and the bell will be rung. If any of the additional character(s) is a command or a buffer control character such as 'erase line', then it will be processed. When sizing the input buffer, it is better to use a size small enough so that the length of the prompt plus the length of the buffer does not exceed the length of a screen line. This can simplify screen maintenance.

ADVANCED APPLICATION . . . Cont.

As one can see, GLINE allows the processing of certain 'control' characters. It also allows only upper-case alpha characters. A modification can be made to allow both upper-case and lower-case alpha characters and even to convert the lower-case to upper-case. It is best to use upper-case because FCS expects upper-case and the testing is simpler.

If an 'erase line' is entered, the entire line is erased and the prompt is re-issued. This is acceptable if the prompt is simple, i.e. one string of text. There are occasions that a prompt may be created by several different routines. In this case, it cannot be regenerated without exiting GLINE. One solution is to erase the line, set the 'Z' flag and test for the 'Z' flag in the calling routine. This allows the calling routine to regenerate the prompt and then call GLINE again. Another approach is to convert an 'erase line' into a series of 'backspaces.' This places the control of the prompt in the calling routine and GLINE does not need the address of the prompt thereby making our routine more versatile. The 'backspace' routine (BACKUP) has been made a subroutine so that this can be easily implemented.

The GLINE routine is listed below with some setup code as an example.

```
CTRLB   EQU     2               ; CONTROL B
CTRLC   EQU     3               ; CONTROL C
BELL    EQU     7               ; BELL
LF      EQU     10              ; LINEFEED
ERASLN  EQU     11              ; ERASE LINE
FF      EQU     12              ; FORMFEED
CR      EQU     13              ; CARRIAGE RETURN
BKSPC   EQU     26              ; BACKSPACE / CONTROL Z
ESC     EQU     27              ; ESCAPE
DEL     EQU     127             ; DELETE/ RUBOUT

;               V2.79/V5.79                     V6.78   V8.79   V9.80
;               ---------------                 ---------  ----------  ----------
LO      EQU     1E27H           ;               3392H   17C8H   17C8H
OSTR    EQU     1E58H           ;               33F4H   182AH   182AH
          :
          :
SETUP:
        LXI     H,BUFFER
        LXI     D,MSG00
        MVI     B,BUFLEN
        CALL    GLINE
        JZ      SETUP           ; NO LINE: TRY AGAIN
        JC      abort routine
        rest of program
          :
          :
BUFFER: DS      128
MSG00:  DB      6,3,ERASLN,'FILENAME>',6,2,239
BUFLEN  EQU     50
          :
          :
;;      GLINE — Get a line from the user
;
;       INPUTS:         HL  => BUFFER
;                       DE  => PROMPT
;                       B   => BUFFER LENGTH
;
;       OUTPUTS:        HL  => BUFFER
;                       A   =  LINE LENGTH
;
;       STATUS:         ⟨Z⟩ – NO LINE
;                       ⟨C⟩ – ABORT
;                       ⟨NZ⟩⟨NC⟩ – GOOD INPUT LINE
GLINE:
        PUSH    H       ; SAVE BUFFER ADDRESS
GLIN02:
```

```
                PUSH    D       ;  SAVE PROMPT ADDRESS
                XCHG
                CALL    OSTR    ;  ISSUE PROMPT
                XRA     A
                STA     CHARIN  ;  FLUSH OUT PREVIOUS CHARACTER
                POP     D
                POP     H
                PUSH    H       ;  SAVE BUFFER ADDRESS
                MOV     C,B     ;  COPY BUFFER SIZE
GLIN04:
                MVI     M,0     ;  SET END OF LINE MARKER
                CALL    CI      ;  READ FROM CONSOLE
                CPI     CTRLC   ;  IS IT CTRL C ?
                JZ      GLX     ;  YES, EXIT FOR CONSL INTERRUPT
                CPI     CR      ;  IS IT CR ?
                JZ      GLX02   ;  YES, GO PROCESS CR
                CPI     BKSPC   ;  IS IT BACKSPACE ?
                JZ      GLIN08  ;  YES, GO PROCESS BACKSPACE
                CPI     ERASLN  ;  ERASE LINE ?
                JZ      GLIN02  ;  GO PROMPT AGAIN
                CPI     ESC     ;  ESCAPE ?
                JZ      GLIN10  ;  IGNORE NEXT CHARACTER
                CPI     DEL     ;  DELETE CHAR ?
                JZ      GLIN08  ;  SAME AS BACKSPACE
                CPI     'Z'+1   ;  IT IS A 'Z' OR LESS ?
                JNC     GLIN04  ;  IF NOT, BAD CHAR
                CPI     ' '     ;  IS IT 'SPACE' OR GREATER ?
                JC      GLIN04  ;  IF NOT, BAD CHAR
                DCR     C       ;  REMAINING BUFFER COUNTER
                JZ      GLIN06  ;  END OF BUFFER: JUMP
                MOV     M,A
                CALL    LO      ;  DISPLAY CHARACTER
                INX     H
                JMP     GLIN04  ;  NEXT CHAR
GLIN06:
                INR     C       ;  BACK UP COUNTER
                MVI     A,BELL
                CALL    LO      :  RING BELL
                JMP     GLIN04
GLIN08:
                MOV     A, C    ;  BUFFER REMAINING
                CMP     B
                JNC     GLIN02  ;  NO CHARACTERS: REPROMPT
                CALL    BACKUP  ;  BACKSPACE ONE
                JMP     GLIN04
GLIN10:
                CALL    CI      ;  NEXT CHARACTER AFTER ⟨ESC⟩
                JMP     GLIN04  ;  IGNORE
GLX:
                POP     H       ;  BUFFER ADDRESS
                ORA     A       ;  CLEAR 'ZERO' FLAG
                STC             ;  SET 'CARRY' FLAG
                RET             ;  CONSOLE INTERRUPT
GLX02:
                MVI     M,0     ;  INSERT TERMINATOR
                POP     H       ;  BUFFER ADDRESS
                MOV     A,B     ;  BUFFER SIZE
                SUB     C       ;  LENGTH OF INPUT
                RET             ;  END 'GLINE'
```

ADVANCED APPLICATION . . . Cont.

ADVANCED APPLICATION . . . Cont.

```
BACKUP:  ; BACKSPACE ONE CHARACTER
         INR     C
         DCX     H          ;  BACKUP BUFFER POINTER
         MVI     A,BKSPC;   'BACKSPACE'
         CALL    LO
         MVI     A,' '      ;  SPACE OVER LAST CHAR
         CALL    LO
         MVI     A,BKSPC;   ANOTHER 'BACKSPACE'
         CALL    LO
         RET
```

If the approach of implementing the 'erase line' as a series of 'backspaces' is desired, then the prompt must be generated external to GLINE and the following changes must be made.

```
GLINE:
         PUSH    H          ;  SAVE BUFFER ADDRESS
GLIN02:
         XRA     A
         STA     CHARIN     ;  FLUSH OUT PREVIOUS CHARACTER
         MOV     C,B        ;  COPY BUFFER SIZE
GLIN04:
           :
           :
         CPI     ERASLN     ;  ERASE LINE ?
         JZ      GLIN12     ;  GO ERASE LINE
           :
           :
GLIN08:
         MOV     A,C        ;  BUFFER REMAINING
         CMP     B          ;  SAME AS BUFFER LENGTH ?
         CC      BACKUP     ;  IF SHORTER, BACKSPACE
         JMP     GLIN04
           :
           :
GLIN12:
         MOV     A,C        ;  REMAINING BUFFER
         CMP     B          ;  SAME AS BUFFER LENGTH ?
         JNC     GLIN04     ;  YES: END OF BACKSPACING
         CALL    BACKUP
         JMP     GLIN12
           :
           :
```

Other changes such as allowing Control H as a 'backspace' can also be easily implemented.

In our next issue, we will begin the discussion of the FCS routines and the associated memory.

REM

system software x-reference

The following is the system software cross reference listing for the COMPUCOLOR II. The listing is for those units with V6.78 system software. In the coming issues of Colorcue, we will publish the listings for both Compucolor II units with more recent software, as well as Intecolor units. This will allow you to take better advantage of your machine's capabilities, and will let you create programs that can be used by everyone, regardless of software version. The companion scratchpad memory locations are found on page 10. Labels referencing RAM locations are denoted in bold print.

LABEL	HEX	LABEL	HEX	LABEL	HEX	LABEL	HEX	LABEL	HEX
A7ON	38E8	B7ON	3A19	BASICI	0052	BCRSY	3A37	BLINK	393F
ACRTSP	0036	BA7OF	3946	BASICW	0040	BEGEX	0038	BRAKE	3AB6
ADDU	2144	BARTX	3D5F	BASOUT	0033	BEGIN	3768	BRATE	3A09
ADHLA	3518	BARTY	3D57	BAUD	0005	BEGOT	3A59	BRATX	3A6F
AESCTB	000B	BARTZ	3D51	BC01	35B2	BEL	3AC3	BREAK	003B
ANHD	351D	BARXM	3C13	BC2BK	35A8	**BFILL**	**81DO**	**BRTRY**	**80E0**
ASCPL	3DFB	BARYM	3C42	BCCIX	3A05	**BHLAD**	**81D4**	BS01	35ED
AUCNT	**81B3**	BASEX	0055	BCHK	3292	BK2BC	35BA	BS02	35F1
AUTOX	0058	**BASFL**	**81F1**	BCHK1	32BB	BKCOL	3928	BS04	365F
B2HEX	33AA	BASICE	0046	BCRSX	3A2A	BLIND	3A09	BS10	237E

LABEL	HEX
BS11	2389
BS12	239D
BS13	236B
BSB01	35C7
BSB08	3652
BSTR	33E9
BUCNT	**81B4**
BUFP	**8047**
BXLOP	3C30
BYLOP	3C67
CARET	000D
CARR	3B4E
CBC	30F5
CBC1	3108
CBC2	3127
CCI	3A09
CCIX	3A01
CD03	2215
CD04	2219
CDHD	211C
CDK	3695
CDMK	002E
CDNM	3691
CDNU	3693
CDRSET	218B
CDSEC	3694
CEN01	2AA5
CENTR	2A9E
CHAIN	3A09
CHDEL	000E
CHDLR	2EFC
CHPLO	39FD
CHTIM	001C
CKEND	26E7
CLOSE	2F26
CLSEQO	3136
CLX	2859
CMASK	**81E0**
CMPDH	3453
CMPHD	344D
CMT1	2AB4
CMT2	2ABF
CMTAB	257A
CODE	3996
CODE2	39A2
COLFL	**81E6**
COLOR	3907
COLW	392C
COMND	0004
COMOF	393B
COMON	393A
COP00	2B20
COP01	2B38
CPLOX	3E03
CPYDV	2C77
CR	3872
CR1	24B3
CR2	24B8
CR3	24BF
CR4	24C3
CRATE	**81E2**
CRC	247D
CRC1	**8043**
CRC2	**8044**
CRCX	249F
CRET	3B56
CRLF	338B
CRSLT	38F0
CRSRT	38B5
CRSUP	38F6
CRSXY	3A09
CRSY	388D
CRTO	0060
CRT1	0061
CRT2	0062
CRT3	0063
CRT4	0064
CRT5	0065
CRT6	0066
CRTCHI	0060
CRTMO	25B7
CRTMSG	25C6
CRTR	0003
CRTRAM	**81AF**
CRTRY	**80E2**
CRTSET	37C0
CRTUBE	396B

LABEL	HEX
CRXDA	006C
CRYDA	006D
CTRK0	**81B1**
CTRK1	**81B2**
CIWO	2C69
CUCNT0	**81B5**
CUCNT1	**81B6**
CURSO	3A0B
D1	358A
D2	359F
DATAM	005A
DBF	**811D**
DBFE	**819D**
DBLK	**811D**
DEL00	29B5
DEL01	29D6
DEL06	2A2D
DEL07	2A3C
DEL08	2A44
DEL09	2A6F
DEL10	2A8A
DELER	2A9B
DELTA	000F
DEV00	2996
DFDV	**80F0**
DFUN	**80F2**
DIG	**3476**
DIP	2D80
DIPS	0006
DIR00	2791
DIR01	2799
DIR02	27C5
DIR03	2816
DIR04	282A
DIR05	27D9
DISPCK	**81BC**
DIVHD	3581
DOWN	3A90
DSBUF	6000
DSBUFS	1000
DSEC	226A
DUP00	2B93
DUP02	2BD9
DUPLX	**81DD**
DX10	2132
DX11	2149
DX12	2159
DX13	2169
DX13A	216E
DX14	21AB
DX14A	21C3
DX15	21D1
DX16	21E6
DX17	21E6
DX18	21F0
DXX	2201
EARLN	3AF8
EARS	3839
EBLF	004B
EBLK	0009
ECFB	000F
ECOP	0045
EDCS	0015
EDEL	003C
EDFN	0024
EDIR	000C
EDRF	002D
EDSY	0012
EDUP	003F
EFCS	3372
EFNF	002A
EFRD	0030
EFWR	0033
EIVC	0003
EIVD	001E
EIVF	0003
EIVP	000F
EIVT	0048
EIVU	0006
ELIN	334E
ELINE	3AEC
EMDV	0018
EMEM	0018
EMESS	262D
EMFN	0015
EMVN	0006
EMVR	001B

LABEL	HEX
ENSA	0021
ENVE	0012
EOFOK	2947
ERALP	3851
ERAS	2B8D
ERASE	3836
ERS1	3631
ERS2	3637
ERSYN	26EB
ERSZ	0039
ESCAP	3A09
ESCAX	3AAA
ESCCRT	**81BF**
ESCD	32C9
ESCDG	32D1
ESCG	32BE
ESCTB	36D8
ESEC	2354
ESIZ	0042
ESKF	000C
ESYN	0009
EVFY	001B
EVOV	0027
EWSZ	0036
EXTBF	**81D6**
EXTIN	0001
EXTOT	0007
F1	25EF
F2	25F2
F3	25F9
F4	2607
FATR	**80F8**
FAUX	**810F**
FBLK	**8115**
FBUF	**8117**
FCS	25EC
FCSEM	262A
FCSEX	2622
FCSFL	81E1
FCSORG	25B7
FCSOUT	3379
FCSX	3301
FDBK	**810D**
FDEN	**810E**
FDRV	**8114**
FEED	3B3D
FERS1	2648
FERS2	265A
FFCN	**8113**
FG0	226C
FG1	226E
FG2	2272
FG3	227F
FG4	2286
FHAN	**8111**
FILL	00FF
FLAD	**8108**
FLBC	**8107**
FNAM	**80F9**
FNEW	0001
FPB	**80F7**
FPBE	**811D**
FPBP	**80F3**
FPROM	0080
FPTR	**811B**
FREE	3A0A
FREEX	3A0A
FSAD	**810A**
FSBK	**8103**
FSIZ	**8105**
FTYP	**80FF**
FULL	3A4E
FVER	**8102**
FXBC	**8119**
G01	3504
GAR1	3258
GAR2	3272
GAREC	3257
GB1	3224
GCMA	3488
GCTRK	256A
GCUCNT	2570
GDATAM	24C8
GDRET	271E
GETBC	37FC
GETTO	2C0C
GH1	22DB

LABEL	HEX
GH2	22E2
GH3	22E9
GH4	22F3
GM01	2CCD
GM02	2CD0
GM03	2CBD
GMPRM	2CA4
GN1D	34E7
GN1Z	34E4
GN2D	34F9
GN2Z	34F6
GNDE	2D86
GODBK	2FB5
GTBYT	322C
GXOUT	2564
HALF	3A4C
HANER	35FC
HDVCT	368E
HER1	22A5
HEX	**81B8**
HOME	386E
HRTR	001E
IDEV	368B
IDM	0055
INCXY	3D2C
INI00	2721
INI01	2761
INI02	2766
INI03	2768
INIBAS	37BD
INITAD	0003
INPCRT	**81C5**
INPFL	**81E3**
INPTB	3728
INSEQO	30E7
INVEC	3BC4
INVY	3BE5
IRBK	318E
IRBKI	3205
IROLL	38C1
IS1	2240
IS2	224D
IS3	2265
IS4	2267
ISEC	2235
ISERL	3974
ISERX	3991
IUNT	368D
IVC	2604
IWBK	318B
IWBKI	3202
JMPD	2F01
JMPHL	3FDD
JUMP	**81E7**
KBCHA	**81FE**
KBDFL	**8IDF**
KBREP	394F
KCHAR	003E
KEDEL	001D
KERDY	001E
KEYBD	3EB3
KEYCO	002B
KEYOT	3A53
KEYTES	0024
KTAB	3FDE
L001	2D94
L002	2DA2
L005	2DC9
L006	2E16
L007	2E30
L008	2E3D
L009	2E40
L010	2E4A
L011	2E5C
L012	2E75
L013	2E82
L014	2F14
L017	2F60
L023	2E7E
L025	2FAA
L1	32E9
L106	2E15
L2	32F7
L3	332A
L4	3334
L5	3349
LADSB	2F03

LABEL	HEX
LBYT	339B
LER1	294E
LER2	2952
LET	346A
LF	38BD
LHXD	33A4
LINBF	**8046**
LINE	3B45
LKC	**81E4**
LL1	3E41
LL10	3EA4
LL2	3E4D
LL3	3E59
LL4	3E65
LL5	3E66
LL6	3E7D
LL7	3E80
LL8	3E8C
LL9	3E8D
LLDA	28CD
LN5X	3897
LN5Y	389E
LO	3392
LOA00	2869
LOCAL	3A4F
LODG	3472
LOFL	**81F9**
LOL1	28C9
LOL3	28E6
LOL4	28EA
LOL5	291F
LOL6	2936
LOL7	293D
LOLDA	288F
LS1	3B93
LS2	3B94
LS3	3B9F
LS4	3BA0
LS5	3BAB
LS6	3BAC
LS7	3BBA
LS8	3BBB
LTIM	0B5A
LTNOR	347E
LTPEN	3B0A
LTYP	2D5A
M1	356A
M2	357A
MASK	0008
MCH01	2C2A
MCH02	2C41
MCHNK	2C1A
MDBLK	**811E**
MFIOA	0000
MODE	0006
MOVDH	343B
MOVHD	3444
MOVXY	3D48
MS1	2CF7
MS150	**81FD**
MS2	2D04
MS3	2D33
MS4	2D42
MST01	3496
MST02	34A3
MST03	34AB
MST04	34AD
MSTR	3495
MULHD	3562
NEGH	3524
NIBL	33B3
NKC	**81E5**
NOCHA	0040
NOLIN	0020
NOROL	3864
NOTH	3525
NROLL	3DA3
NSEC	000A
NTRK	0029
NTYP	2D53
OBC	**80E3**
OCODE	**80F5**
ODDFL	**81EE**
OFFPB	26E5
OPDIR	2D60
OPEN	2DAB
OPENX	2C86

REM Cont.

REM Cont.

LABEL	HEX
OPENY	2C89
OPOX	2C00
OPX01	2CA2
ORAM	**80F0**
ORCHA	3D21
ORHD	352C
OSEC	**80ED**
OSERL	3A61
OSTR	33F4
OSTR1	3404
OSTR2	340A
OSTR3	340C
OSTR4	3410
OSTR5	3418
OUTBL	3718
OUTCRT	**81C2**
OUTFL	**81F8**
OUTHL	**81FB**
OVERS	**80F6**
P2SNUM	34D2
PAGE	3A86
PB1	3237
PBINX	3C05
PBINY	3C8B
PBYT	34CD
PCFSP	3087
PCOLN	34B8
PCRAD	**81D8**
PDV	2FDE
PDV01	2FF3
PDV02	3002
PDV03	300F
PDV04	3016
PDV05	3019
PDV06	3026
PDV07	3055
PDV08	3062
PFS01	30A5
PFS02	30AB
PFS03	30BB
PFS04	30C1
PFSPC	3077
PLINC	3C7C
PLOFL	**81DA**
PLOKB	3DE9
PLOTX	3CB2
PLPTY	3BFD
PLTAB	3D6F
PNFSP	306E
PNUM	34D8
POTON	3D99
POUND	2511
PPFSP	3074
PRDEV	2CD4
PRINT	3A96
PRM0	0080
PRM1	0081
PRM2	0082
PRM3	0083
PRM4	0084
PRM5	0085
PRM6	0086
PRMPT	3382
PROCES	398C
PROLL	0007
PSBYT	34CA
PSFSP	3068
PSNUM	34D5
PSPAC	34B3
PSSTR	34BD
PSTAT	**81DB**
PSTR	34C0
PTBYT	324A
PTREC	3285
PTYP	2D57
PUP	**81B7**
PUTEZ	3DE6
PUTXD	3C9B
PUTXZ	3DC6
PUTYD	3CEB
PUTYZ	3DCD
PUTZZ	3DAD
PVREC	327B
PWRUP	37B1

LABEL	HEX
PXYCH	3D13
PXZER	3BF3
Q01	3EB8
Q02	3EBC
Q03	3EC9
Q04	3ECB
Q05	3EE7
Q06	3EEF
Q07	3EF7
Q08	3F07
Q09	3F2C
Q10	3F3F
Q11	3F43
Q12	3F54
Q13	3F81
Q14	3F85
Q15	3F9E
Q16	3FAC
Q20	3FC2
Q21	3FCB
QLDA	2AAB
QTYP	2AAE
RATEA	0016
RATEB	002F
RBLK	3182
RBLKI	31F9
RBYTE	246A
RBYTEC	2474
RCOMD	0017
RD	2EFB
RD00	2411
RD01	241C
RD02	2429
RD03	2435
RD04	2438
REA00	26F1
READ	2EA3
READY	**81FF**
REN00	2AC1
RERR	2453
RESET	26A5
RF1	2880
RF2	2883
RFLG	**80E1**
RGAPS	0003
ROLDA	0066
ROLFL	**81DC**
ROLL	3A85
ROLLN	**81CD**
ROT	3EFE
RRTR	000A
RS01	26AB
RS02	26CE
RST1J	**81C8**
RSTBF	0002
RTST	33D5
RTST2	33D8
RUN00	2956
RUN01	296D
RUN02	297C
RWB1	31EC
RWB2	31EF
RWBCM	318F
RWE	2EC5
RWICM	3206
RWSEQI	30C6
RXBUF	0000
RXSER	0020
S1OUT	33C3
SAV00	2833
SAVE	3FD0
SBC	**8042**
SBCHA	39BC
SCMN	3554
SEC	**80EE**
SEC15	**81D7**
SEEK	2222
SEEKF	22AA
SFR	2F96
SHIFF	3B85
SHLHD	353A
SHRHD	3544
SL1	353E
SOK	0000
SP64C	39B4
SPNOR	3460
SR1	3548

LABEL	HEX
SRTR	001E
SSIDA	0008
SSOSE	0010
STACK	**8042**
START	0000
STARTI	006E
STARX	376C
STAT5	0003
STEPCD	0027
STEPS	2525
STIM	003F
STOPIT	006A
STOR1	39EB
STOR2	39F1
STP1	2529
STP2	2536
STP3	2558
STPIN	253D
STPOUT	254D
STPTIM	0014
STPWAT	255E
STWO	2C67
STYP	2D5D
SUBHD	3459
SUBU	214D
SULD	2986
SVCHA	39DC
SVCRS	3873
SX0	2502
SX01	250C
SYSORG	211C
TAB	38B1
TAU	0005
TBADDR	368B
TBC	**80EB**
TBL00	36A8
TBL24	36C8
TBLK	**80E7**
TDRV	**80E6**
TEMP0	**81F2**
TEMP1	**81F3**
TEMP2	**81F4**
TEMP3	**81F5**
TEMP4	**81F6**
TEMP5	**81F7**
TEMPHL	**80DE**
TESHI	39CE
TEST	3A09
TESTB	3A45
TESTC	3A2D
TESTX	3831
TFCN	**80E5**
TFILE	0002
TFREE	0001
THRUFL	**81DE**
TIM3X	3EA6
TIM4X	3AD1
TIM4Y	3AE0
TIM4Z	3AE6
TIME1	0009
TIME2	000A
TIME3	000B
TIME4	000C
TIME5	000D
TIMX1	0000
TIMX2	0008
TIMX3	0018
TIMX4	0030
TIMX5	0038
TMEM	**80E9**
TMODE	39FE
TMP1	**81AB**
TOFF	2154
TPROG	001B
TPROT	0001
TRAM	**80DE**
TRK	**80EF**
TWOUT	3B25
TXBUF	0006
TXCONT	3B1D
TXOUT	3B30
TXPEN	3B5D
TXSER	0028
UPDATE	3805
UPTIM	0640
VCRAD	**81CB**
VCRSY	3A1F

LABEL	HEX
VE00	2305
VE01	2322
VE02	232A
VE03	2336
VE04	233D
VECT0	3E2D
VECTY	3BBE
VERR	230B
VFILL	**81CE**
VHLAD	**81D2**
VISIB	3A0A
VRTR	0002
VTP	24DD
VTP1	24EC
VTP2	24F4
WATL	3429
WATS	341C
WB1	245F
WBLK	317F
WBLKI	31F6
WBYTE	245E
WF2	284F
WIG1	22BA
WIG2	22C4
WIG3	22D2
WL1	342A
WL2	242D
WR	2EF8
WR00	23A6
WR01	23BF
WR02	23C4
WR03	23D0
WR04	23D4
WR05	23E6
WR06	23F0
WRDIR	2F75
WRI00	26EE
WRITE	2ECC
WRTR	0004
WS1	341F
WXYZ	2573
X80	7000
XDATA	**81EC**
XFBLK	**81A1**
XFBUF	**81A3**
XFDRV	**81A0**
XFER	219A
XFFCN	**819F**
XFHAN	**819D**
XFXBC	**81A5**
XINTR	0010
XORHD	3533
XOUT0	**81AF**
XOUT1	**81B0**
XTWO	**81EA**
XYMIT	3B13
XYTAB	3D67
XZERO	**81EF**
YDATA	**81ED**
YTWO	**81EB**
YZERO	**81F0**
ZERFL	39A9
ZFATR	**8083**
ZFAUX	**809A**
ZFBLK	**80A0**
ZFBUF	**80A2**
ZFDBK	**8098**
ZFDEN	**8099**
ZFDRV	**809F**
ZFFCN	**809E**
ZFHAN	**809C**
ZFLAD	**8093**
ZFLBC	**8092**
ZFNAM	**8084**
ZFPB	**8082**
ZFPBE	**80A8**
ZFPTR	**80A6**
ZFSAD	**8095**
ZFSBK	**808E**
ZFSIZ	**8090**
ZFTYP	**808A**
ZFVER	**808D**
ZFXBC	**80A4**
ZH	355E
ZPTR	30D9
ZRAM	**8082**
ZZZZZZ	3FFA

REM

the "call" function

Myron Steffy of Sun City, Arizona, has been an active COMPUCOLOR II user since the "early days". He has been a prolific correspondent and his comments and suggestions are always appreciated. Myron submitted the following article which illustrates the use of the CALL function to load a screen display from RAM. His program is concise and should be easy for most readers to understand.

"For a long time I have been trying to find a useful application for the "call" function in Basic. Other than its use with the "Soundware" device, I don't recall anything having appeared in Colorcue. One of the first programs I originated in Basic was a satellite tracking routine for the series of amateur radio satellites generically known as "OSCAR". Not too long ago I was trying to show the satellite's motion graphically on a map of the United States. At first I used an erasing subroutine that worked well but left no trail. Then I decided that it would be better to leave each track in place for a full day's run.

The next thing that cropped up was the necessity for recalling the map of the U.S. each time. This particular graphic was stored on disc as a "Screen.Dsp" which required accessing the disc drive every few seconds. This offended my sense of propriety as I had all of this memory just sitting there. To put the rather elaborate graphic into code was a task that I wasn't up to. This is the occasion for the use of the "call" function.

The idea briefly is this: store the screen display in high memory and then move it into screen memory 'en block' with a machine language subroutine to be accessed by the call function. Strangely enough, it is the first piece of assembly language that I have ever written that worked the first time. This I must publish!

The source file is attached as well as a Basic routine that will enter the machine code. The procedure for calling it up is simply "X = CALL ⟨0⟩". The display is recalled within one or two seconds without bringing up the disc drive. I can think of a number of uses for it — recalling a checker board or any other display that will be used over and over. Saves a lot of wear and tear on the machinery.

The disc contains the Screen.Dsp (courtesy of the Northern California Users Group); the source file for "Reload" and its PRG. version; then you will find the Basic routine for "Reload" and a program called "Demo" which illustrates the use of the call function in the program.

Referring to the "Reload" program in Basic, line 200 samples the display at three points to see if it has already been loaded. If not, line 210 performs this function, placing it at the top end of a 16K memory. The "Peeks" would have to be changed for the particular display in use. Line 200 could be eliminated if subsequent "runs" were started at line 250. Lines 220 to 250 load the machine code for the call routine. This is probably faster than loading the assembly version from FCS. Line 260 inserts the call jump address which will be lost if "CPU Reset" is used. Line 280 clears the screen, fixes the "page" mode, homes the cursor and executes the "Call Jump".

As you will see from the source file, the program is structured to run on either 6.78 or 8.79 Basic and can be used anywhere it is necessary to frequently recall a display. I have set it up for 16K although of course it could be readily pushed up to the top of a 32K RAM. I hope that it will be of some use to the Colorcue readers."

CALL ROUTINE TO LOAD SCREEN DISPLAY FROM RAM
by Myron T. Steffy, Sun City, AZ 11/28/80
for 6.78 or 8.79 Basic

SCREEN DISPLAY IS TO BE LOADED AT 0AFFFH BY MAIN PROGRAM
BASIC PROGRAM MUST INCLUDE THE FOLLOWING STATEMENT:

"POKE 33283,33:POKE 33284,175:REM CALL JUMP"

TO CALL UP DISPLAY, USE "PLOT 8: X = CALL ⟨0⟩"

```
START:   ORG    0AEFFH      ; 44799
         LXI    SP,STACK
         CALL   SETUP       ; WHICH BASIC?
         LXI    H,RELOAD
         SHLD   8203H       ; CALL JUMP
         MVI    A,0C3H      ; JUMP
         STA    81BFH       ; ESCAPE ↑- 33215
         LXI    H,RELOAD
         SHLD   81C0H       ; 33216
         LXI    H,0AEFEH    ; PROTECT MACHINE LANGUAGE
         SHLD   80ACH       ; 32940
         MVI    A,45H       ; 'E' EXIT TO BASIC
         JMP    EXIT

RELOAD:  PUSH   PSW         ; SAVE STATUS
```

REM Cont.

```
            PUSH    H               ;  SAVE H & L
            LXI     H,0AFFFH        ;  LOW END OF DISPLAY
            LXI     D,7000H         ;  SCREEN ADDRESS
            LXI     B,0BFFFH        ;  HIGH END

NEWAD:      MOV     A,M             ;  FETCH CONTENTS TO BE MOVED
            STAX    D               ;  STORE IN NEW LOCATION
            MOV     A,H             ;  HIGH BYTE OF 'FROM' ADDRESS
            CMP     B               ;  PAGE LIMIT?
            JNZ     INCAD           ;  NO, CONTINUE TRANSFER
            MOV     A,L             ;  YES, GET LOW BYTE
            CMP     C               ;  LOW ADDRESS LIMIT?
            JZ      GHOME           ;  ALL FINISHED

INCAD:      INX     H               ;  NO, ADVANCE 'FROM' POINTER
            INX     D               ;  ADVANCE 'TO' POINTER
            JMP     NEWAD

GHOME:      POP     H               ;  RESTORE H & L

            POP     PSW             ;  AND STATUS
            RET                     ;  RETURN TO CALLING PROGRAM

EXIT:       JMP     0000H           ;  TO BASIC

SETUP:      LDA     0001H
            CPI     6CH
            JNZ     VER879
VER678:     LXI     H,053AH         ;  EXIT
            SHLD    EXIT+1
            RET
VER879:     LXI     H,2420H         ;  EXIT
            SHLD    EXIT+1
            RET

            DS      20H             ;  STACK AREA
STACK:
            END     START
```

```
100  REM ROUTINE FOR LOADING A SCREEN DISPLAY INTO HIGH MEMORY
110 REM
120 REM  AND RECALLING IT TO THE SCREEN WITH A CALL STATEMENT.
140 REM
160 REM
200 IF PEEK (45963) + PEEK (47198) + PEEK (48225) = 558THEN 220
210 PLOT 27,4:PRINT "LOAD SCREEN.DSP AFFF":PLOT 27,27
220 DATA 245,229,33,255,175,17,0,112,1,255,191,126,18,124
230 DATA 184,194,56,175,125,185,202,61,175,35,19,195,44,175
240 DATA 225,241,201,0,0,0
250 FOR AD = 44833TO 44865:READ VL:POKE AD,VL:NEXT AD
260 POKE 33283,33:POKE 33284,175:REM CALL JUMP ADDRESS
270 RESTORE
280 PLOT 12,27,24,3,0,0:X = CALL (0)
```

REM

key scratchpad memory locations

These locations are offered in conjunction with the system software cross reference article which starts on page 6. This is the system RAM reference listing with decimal value and description.

LABEL	HEX	DECIMAL	DESCRIPTION
BASFL	81F1	33265	BASIC output FLAG
BFILL	81DO	33232	Blind fill (+0= A7 bit, +1= CCI)

LABEL	HEX	DECIMAL	DESCRIPTION
BHLAD	81D4	33236	Blind cursor H&L address
BRTRY	80E0	32992	Block reentry counter
BUCNT	81B4	33204	Spare
BUFP	8047	32839	FCS line buffer
CMASK	81E0	33248	Current mask register setting
COLFL	81E6	33254	Flag (FG/BG) 0=off, 1=on
CRATE	81E2	33250	Current baud rate setting
CRC1	8043	32835	1st CRC byte count for disk
CRC2	8044	32836	2nd CRC byte count for disk
CRTRAM	81AF	33199	CRT RAM
CRTRY	80E2	32994	'Chunk' reentry counter
CTRK0	81B1	33201	Current track Micro Drive 0
CTRK1	81B2	33202	Current track Micro Drive 1
CUCNT0	81B5	33205	User count Micro Drive 0
CUCNT1	81B6	33206	User count Micro Drive 1
DBF	811D	33053	Directory block buffer
DBFE	819D	33181	End of directory block buffer
DBLK	811D	33053	'This' directory block number
DFDV	80F0	33008	Default device (ASCII)
DFUN	80F2	33010	Default unit (ASCII)
DISPCK	81BC	33212	Jump to display clock
DUPLX	81DD	33245	Duplex FLAG => (0 = local, − = full, + = half)
ESCCRT	81BF	33215	User ESCAPE ∧ jump vector
EXTBF	81D6	33238	External output port buffer
FATR	80F8	33016	Attribute byte
FAUX	810F	33039	New file closing size, or aux. byte count for sequential routines.
FBLK	8115	33045	Block number for transfer
FBUF	8117	33047	Buffer pointer for transfer
FCSFL	81E1	33249	FCS output FLAG
FDBK	810D	33037	Directory block number
FDEN	810E	33038	Directory entry number
FDRV	8114	33044	Drive number
FFCN	8113	33043	Handler function code
FHAN	8111	33041	Handler address
FLAD	8108	33032	Load address for 'image' file
FLBC	8107	33031	Byte count of last block
FNAM	80F9	33017	File name
FPB	80F7	33015	Open type code
FPBE	811D	33053	End of system FPB
FPBP	80F3	33011	File parameter block pointer
FPTR	811B	33051	Buffer pointer for sequential routines
FSAD	810A	33034	Start address for 'image' file
FSBK	8103	33027	Starting block number
FSIZ	8105	33029	Number of blocks
FTYP	80FF	33023	File type
FVER	8102	33026	File version number
FXBC	8119	33049	Byte count for transfer
HEX	81B8	33208	Binary fractions of a second
	81B9	33209	0 to 59 seconds of real time clock
	81BA	33210	0 to 59 minutes of real time clock
	81BB	33211	0 to 23 hours of real time clock
INPCRT	81C5	33221	User input FLAG jump vector
INPFL	81E3	33251	Serial input FLAG
JUMP	81E7	33255	Jump used for cursor position — left, right, etc.
KBCHA	81FE	33278	Keyboard character
KBDFL	81DF	33247	Keyboard FLAG
LINBF	8046	32838	BASIC Line buffer
LKC	81E4	33252	Last key code
LOFL	81F9	33273	System output FLAG
MDBLK	811E	33054	Maximum directory block number
MS150	81FD	33277	Counter for 150 millisecond delay
NKC	81E5	33253	New key code

REM Cont.

LABEL	HEX	DECIMAL	DESCRIPTION
OBC	80E3	32995	Old byte count
OCODE	80F5	33013	Open type code
ODDFL	81EE	33262	
ORAM	80F0	33008	FCS RAM
OSEC	80ED	33005	Old sector number
OUTCRT	81C2	33218	User output FLAG jump vector (table 6)
OUTFL	81F8	33272	Output port FLAG
OUTHL	81FB	33275	Output port H&L address
OVERS	80F6	33014	Original version
PCRAD	81D8	33240	Plot cursor address
PLOFL	81DA	33242	Current plot submode
PSTAT	81DB	33243	
PUP	81B7	33207	Power up FLAG
READY	81FF	33279	Keyboard character ready flag
RFLG	80E1	32993	'Restore' FLAG / counter
ROLFL	81DC	33244	Roll FLAG => (0= no roll, 1 = roll)
ROLLN	81CD	33229	Roll count (0 = no roll)
RST1J	81C8	33224	Timer 2 jump vector
SBC	8042	32834	Sector byte count for disk
SEC	80EE	33006	Sector number
SEC15	81D7	33239	Repeat key scan counter
STACK	8042	32834	Stack from screen to here
TBC	80EB	33003	Byte count
TBLK	80E7	32999	Block number
TDRV	80E6	32998	Drive number
TEMP0	81F2	33266	Temporary
TEMP1	81F3	33267	Temporary
TEMP2	81F4	33268	Temporary
TEMP3	81F5	33269	Temporary
TEMP4	81F6	33270	Temporary
TEMP5	81F7	33271	Temporary
TEMPHL	80DE	32990	Free for future use
TFCN	80E5	32997	Function code
THRUFL	81DE	33246	
TMEM	80E9	33001	Memory buffer pointer
TMP1	81AB	33195	Used by COPY & maybe others?
TRAM	80DE	32990	Temporary RAM start in BASIC RAM
TRK	80EF	33007	Track number
VCRAD	81CB	33227	Visible cursor address (+0 = X, +1 = Y)
VFILL	81CE	33230	Visible fill (+0 = A7 bit, +1 = CCI)
VHLAD	81D2	33234	Visible cursor H&L address
XDATA	81EC	33260	Plot mode temporary
XFBLK	81A1	33185	Auxiliary block buffer
XFBUF	81A3	33187	Auxiliary buffer pointer
XFDRV	81A0	33184	Auxiliary drive number
XFFCN	819F	33183	Auxiliary handler function code
XFHAN	819D	33181	Auxiliary handler address
XFXBC	81A5	33189	Auxiliary byte count
XOUT0	81AF	33199	Current phase Micro Drive 0
XOUT1	81B0	33200	Current phase Micro Drive 1
XTWO	81EA	33258	Plot mode temporary
XZERO	81EF	33263	Plot mode temporary
YDATA	81ED	33261	Plot mode temporary
YTWO	81EB	33259	Plot mode temporary
YZERO	81F0	33264	Plot mode temporary
ZFATR	8083	32899	Auxiliary FCS FPB storage
ZFAUX	809A	32922	Auxiliary FCS FPB storage
ZFBLK	80A0	32928	Auxiliary FCS FPB storage
ZFBUF	80A2	32930	Auxiliary FCS FPB storage
ZFDBK	8098	32920	Auxiliary FCS FPB storage
ZFDEN	8099	32921	Auxiliary FCS FPB storage
ZFDRV	809F	32927	Auxiliary FCS FPB storage

LABEL	HEX	DECIMAL	DESCRIPTION
ZFFCN	809E	32926	Auxiliary FCS FPB storage
ZFHAN	809C	32924	Auxiliary FCS FPB storage
ZFLAD	8093	32915	Auxiliary FCS FPB storage
ZFLBC	8092	32914	Auxiliary FCS FPB storage
ZFNAM	8084	32900	Auxiliary FCS FPB storage
ZFPB	8082	32898	Auxiliary FCS FPB storage
ZFPBE	80A8	32936	End of auxiliary FPB, End of BASIC input buffer
ZFPTR	80A6	32934	Auxiliary FCS FPB storage
ZFSAD	8095	32917	Auxiliary FCS FPB storage
ZFSBK	808E	32910	Auxiliary FCS FPB storage
ZFSIZ	8090	32912	Auxiliary FCS FPB storage
ZFTYP	808A	32906	Auxiliary FCS FPB storage
ZFVER	808D	32909	Auxiliary FCS FPB storage
ZFXBC	80A4	32932	Auxiliary FCS FPB storage
ZRAM	8082	32898	FCS stuff / BASIC input buffer

REM

system software map

RESTART VECTORS INITIAL VALUES	EXT'D DISK BASIC ROM	FILE CONTROL SYSTEM ROM	CRT & PLOT GRAPH ROM	AVAILABLE ROM SPACE FOR USER FIRMWARE	SCREEN REFRESH RAM HIGH/LOW	SYSTEM SCRATCH PAD RAM	USER RAM
0000	0040	211C	36AB	4000	6000	8000	8200
to	to	to	to	to	to	to	to
003F	211B	36AA	3FFF	5FFF	7FFF	81FF	FFFF

NOTE: In BASIC 8000-8299 is System and BASIC Scratch pad RAM.
In BASIC 829A-FFFF is the actual user RAM available.
To use 4000-5FFF, an ADD-ON ROM STACK is required. (CC P/N 100980)

CORRECTION

handshake modification

In a recent issue of Colorcue we published a handshake modification. Unfortunately, that data was incorrect and caused a few problems. Below find the proper information:

1. Tie Pin 9 of the J2 edge connector to UD1 Pin 4.
2. Tie Pin 6 of UD1 to Pin 3 of UC1.
3. Tie Pin 4 of UC1 to Pin 10 of UE1.
4. Add a 10K 1/4 W resistor between Pin 4 of UD1 to +12VDC.

INTELLIGENT SYSTEMS ANNOUNCES:

the 3650 series

The Intecolor 3650 series of terminals and desktop computers, which was released in October, 1980, provides a cost-effective solution for the small business requiring good capabilities at a low price. The 3650 preserves many of the features of the 3621, while adding some design improvements that have upgraded the performance. For example, the logic board is of a completely new design, and the disk controller is of high computer grade. The 3650 has a built-in 90K bytes mini-disk (instead of the 3621's micro) and offers the option of add-on disks such as dual-sided double 8-inch floppies as well as hard disk. The internal drive has been specially tested to ensure the kind of reliability that is imperative in serious applications. For those users who already have significant software on micro-disk, upgrading to the mini-disk is no problem — there's a utility to transfer programs to the 3650, and almost all BASIC programs will run without modification. Assembly language programs will need some slight changes. For more information, contact our customer service department.

IT'S GRAPHIC!

bar graph and scaler

We've decided to add a graphics column to Colorcue because so many of you have questions about using the graphics, and because, after all, graphics is one of the CCII's major features. While you can easily find books that teach assembly language or 'DO' loops or ASCII codes, finding written information about graphics is a little more difficult. And even when you do get information, it may not be specialized to the CCII, so we thought we'd try to help

IT'S GRAPHIC! Cont.

out through Colorcue.

If you will recall the first two issues of Colorcue, they did include graphics information. We had the now-famous 'Random Rectangles' and the less ubiquitous 'Circular Plots'. This month we're going to start where they left off and explain another simple feature of the CCII's graphics — bar graphs. The program below creates a bar graph and automatically scales it to reflect the data given for the graph. All changes necessary for Intecolor equipment are contained in REM statements.

Line 100 of the program erases the screen with foreground green and background black. Lines 110 and 120 contain the data for the scaling factor. Line 130 dimensions the variables MR, SF, and SP, which stand for Maximum Range, Scaling Factor, and Scaling Pointer. Line 140 is a loop to read this data. Line 150 initializes three variables.

Line 160 generates a random number between 12 and 6. Line 170 generates a minimum and maximum bar value. Line 200 sets everything in readiness for drawing the bar graph. It sets the page mode, erases the page, and draws the x and y axes of the graph. Lines 210 through 280 print variables on the screen.

Line 300 sends program control to line 470 to get the scaling factor. Lines 330 through 390 generate the dashed line for the graph routine. Line 400 defines the y1 and y2 variables.

Line 410 sets the foreground to magenta and draws a vertical bar, as does line 420.

Line 480 starts the subroutine that contains the scaler for the program, with line 520 determining the scaling factor.

```
 90 REM BAR GRAPHS AND SCALER
100 PLOT 6,2,12
110 DATA 10,1,3,15,2.5,5,20,2.5,4,25,2.5,3,30,5,5
120 DATA 40,5,4,50,5,3,60,10,5,80,10,4,100,10,3
130 DIM MR(10),SF(10),SP(10)
140 FOR I=1 TO 10 : READ MR(I),SF(I),SP(I) : NEXT I
150 YO=2 : YX=0 : YI=999
160 R=RND(1)*12-6 : R=10^R
170 MI=-R+RND(1)*2*R : MX=MI+RND(1)*R
180 PLOT 27,88,15,6,2,12
190 X=127 : Y=127 : REM ON INTECOLOR 8001 USE X=159 : Y=191
200 PLOT 2,X,0,242,20,0,20,Y,255
210 PLOT 3,45,20 : PRINT "MAX = ";MX
220 PLOT 3,45,21 : PRINT "MIN = ";MI
230 PLOT 3,45,23 : PRINT " YI = ";YI
240 PLOT 3,45,24 : PRINT " YX = ";YX
250 PLOT 3,45,26 : PRINT " SI = ";SI
260 PLOT 3,45,27 : PRINT " SX = ";SX
270 PLOT 3,45,29 : PRINT " BI = ";BI
280 PLOT 3,45,30 : PRINT " BX = ";BX
290 Y=31 : REM ON INTECOLOR 8001 USE Y=47
300 GOSUB 470
310 BT=M1
320 FOR I=1 TO 12 : IF Y<0 THEN I=12 : GOTO 390
330 PLOT 3,0,Y,
340 IF INT(ABS(M1/SF))<1 THEN M1=0
350 PLOT 19 : PRINT RIGHT$("          "+STR$(M1),9)
360 PLOT 6,4 : IF M1=0 THEN PLOT 6,7
370 IF I>1 THEN PLOT 3,11,Y : PRINT "----------------------------------------------------"
380 M1=M1+SF : Y=Y-SP(KK)
390 NEXT I
400 Y1=(MI-BT)*SP(KK)*4/SF : Y2=(MX-BT)*SP(KK)*4/SF
410 PLOT 6,5,2,40,YO+Y1,242,40,YO+Y2,255
420 PLOT 6,5,2,41,YO+Y1,242,41,YO+Y2,255
430 IF Y2-Y1<YI THEN YI=Y2-Y1 : SI=MI : SX=MX
440 IF Y2-Y1>YX THEN YX=Y2-Y1 : BI=MI : BX=MX
450 IF I$="A" THEN 230
460 GOTO 160
470 REM ** SCALE **
480 N9=MX-MI : L9=LOG(ABS(N9))/LOG(10)
490 D9=INT(L9)-1 : M9=SGN(N9)*INT(10^(L9-D9)+.999)
500 FOR II=1 TO 10 : IF M9<=MR(II) THEN KK=II : II=10
510 NEXT II
520 IF KK>10 THEN KK=1 : D9=D9+1
530 SF=SF(KK)*10^D9 : M1=INT(MI/SF)*SF
```

```
540  M2=MX : IF MX/SF< > INT (MX/SF) THEN M2=INT(MX/SF) *SF+SF
550  IF (M2-M1)>(MR(KK)+.001) *10^D9 THEN KK=KK+ 1  : GOTO 520
560  RETURN
```

layered design

The following is another simple program that uses the COMPUCOLOR II's graphics capabilities. It draws a layered design in various colors. Changes required to run this program on Intecolor systems are given in REM statements.

```
100  REM OVERLAYING GRAPHIC DESIGN
110  RT=RND(10*RND(34))
120  PLOT 6,0, 12, 15,27,88
130  XX=122 : YY=122 : REM FOR INTECOLOR 8001 USE XX=154 : YY=186
140  X=120 : Y=120 : REM FOR INTECOLOR 8001 USE X=152 : Y=184
150  PLOT 3,0,0,6,2
160  INPUT "HOW MANY LAYERS? (TRY 3) ";LC : PLOT 28, 11
170  BG$="N"
180  INPUT "STEP SIZE? (TRY 14)";S : PLOT 28, 11
190  REM
200  AC=AC+1 : IF AC>=LC THEN AC=0 : FOR XD=1 TO 1000 : NEXT : PLOT 12
210  GOTO 400
220  X1=XA : Y1=YA : X2=XB : Y2=YB : X3=XC : Y3=YC : X4=XD : Y4=YD
230  A=INT(7*RND(11)+1)
240  IF BG$<>"N" THEN PLOT 12
250  XP=X1 : YP=Y1 : XQ=X2 : YQ=Y2
260  PLOT 29, 16+A
270  S1=(X3-X1)/S
280  S2=(X4-X2)/S
290  S3=(Y3-Y1)/S
300  S4=(Y4-Y2)/S
310  FOR XP=X1 TO X3 STEP S1
320  PLOT 2,XP,YP,242,XP,YP,XQ,YQ,255
330  PLOT 2,XX-XP,YY-YP,242,XX-XP,YY-YP,XX-XQ,YY-YQ,255
340  PLOT 2,XX-XP,YP,242,XX-XP,YP,XX-XQ,YQ,255
350  PLOT 2,XP,YY-YP,242,XP,YY-YP,XQ,YY-YQ,255
360  YP=YP+S3 : XQ=XQ+S2
370  YQ=YQ+S4
380  NEXT XP
390  GOTO 190
400  XA=INT(X*RND(1)+2)
410  YA=INT(Y*RND(1)+2)
420  XB=INT(X*RND(1)+2)
430  YB=INT(Y*RND(1)+2)
440  XC=INT(X*RND(1)+2)
450  YC=INT(Y*RND(1)+2)
460  XD=INT(X*RND(1)+2)
470  YD=INT(Y*RND(1)+2)
480  GOTO 220
```

NEW PRODUCTS

comp-u-writer

If you still consider word processing a function strictly for the office, then you haven't been keeping up with the progress that's been made in this field. WP (word processing) systems are used in all kinds of applications. The number of word processing systems available in today's market is exceeded only by the number of stars in the sky, and yet each of these systems has features and capabilities all its own. At ISC, we have two word processing systems. One is a CP/M system for the Intecolor Business Systems, on which development started about two years ago. Colorcue has been created and printed using the word processor from the very first issue. Now available for COMPUCOLOR II owners is a word processing system that allows you to create, edit, and update documents of all kinds.

The system is "COMP-U-Writer", and it was designed especially for the COMPUCOLOR II. COMP-U-Writer uses color effectively to make learning and using the system easier.

The COMP-U-Writer compares very favorably with other WP systems available for microcomputers. It has many of the sophisticated features found on expensive stand-alone systems. COMP-U-Writer lets you generate

NEW PRODUCTS . . . Cont.

text on the screen, and then allows you to make corrections or formatting changes with a few keystrokes. When the copy reads exactly as you want it, you send the file through the RS232C port to a printer. With COMP-U-Writer, you avoid all the typing, erasing, and retyping required with conventional typewriting. COMP-U-Writer uses function keys to access its many capabilities, such as:

search and replace	center
move	delete
new page	

COMP-U-Writer can be useful for the one-man business, but it has many uses for other computer users as well. Students can write term papers on the COMP-U-Writer, and save time by obviating all the rewriting and retyping normally needed. Many Compucolor II users find COMP-U-Writer invaluable for personal correspondence as well.

We are pleased to enter the WP market with COMP-U-Writer, because we believe that it is a good system. It was reviewed in InfoWorld a few months ago and received a very good recommendation. The system was evaluated for:

Functionality	Good
Ease of Use	Excellent
Documentation	Excellent
Error Handling	Fair
Support	Excellent

The COMPUCOLOR II is ideal for WP because it has a commercial quality keyboard which so many of its competitors lack. And color enhancement allows improved communication and operator response. COMP-U-Writer sells for $262.50 and can be ordered from your dealer or from our factory. COMP-U-Writer requires at least 16K and a 117-key keyboard.

ink-jet printer

PrintaColor Corporation of Norcross, GA, announces the introduction of their IS8001 Color Ink-Jet Printer. Designed primarily for graphics applications, the IS8001 can print in seven colors (yellow, magenta, cyan, blue, green, red, and black) on a white background. The IS8001 contains its own microcomputer, including 16K RAM used as a data buffer, Since the printer is "intelligent", it operates with a minimal burden on the host computer. The host's processing ability is not tied up except for the 8-10 second initial transmission of data to the printer.

The unit has 12 ink-jet nozzles, four for each of the three primary colors. Additional colors are made by overlaying the primaries. Resolution is 90 dots per inch. The paper system is continuous-feed, Z-fold and 14⅞ inches wide with 80 characters per line capability.

Initial models of the IS8001 are compatible with the Intecolor 8001 series computers and terminals. Also available are models compatible with the ISC 3600 series and the COMPUCOLOR II. The price of the unit is $6000.00.

With its easier readability and additional computer functions, the Printacolor IS8001 can effect considerable time savings and higher efficiency for the color CRT user.

For additional information, write to Printacolor Corporation, P.O. Box 52, Norcross, GA 30071, or call (404) 448-2675.

NUTS AND BOLTS

compucolor bell

Those with the version 8.79 software have available the option of installing a bell on their COMPUCOLOR IIs. As you know, there was no provision for a bell in the original design of the machine, but because a bell can be quite useful, there is now a way to attach this device to the computer.

First, and most obviously, the bell is great for punctuating computer programs that require user input. You can program the bell to sound when a mistake is made or when a response is required. The bell makes these programs more interactive because it focuses user attention at critical moments and disallows error. In real-time applications, the bell can add excitement with sound effects or indicate a time out.

Secondly, and this use is one you might not have thought of, the bell is very valuable in the debugging process. PLOT 7 is the command sequence that rings the bell, and by inserting PLOT 7's at various points in a program, you can determine if the program is passing certain statements. Or, if the program has to perform a given function a specified number of times, you can insert a PLOT 7 and audibly count to see if it's successful.

Assembly and installation of the bell is not especially difficult, but it requires a little bit of time. You will need:

Sonalert assembly	soldering iron
sponge, etc.	needlenose pliers
wire cutters	60/40 rosin core solder
IN914 diode	mounting bracket or glue
16-pin socket	
lt. gauge (24-28 AWG) insulated stranded wire — two 12" lengths, one red, one black	

The Sonalert assembly can be purchased from Compucolor Corporation. Order part number #010015. The price is $21.00. Or, you may be able to find the Sonalert at radio supply/hobby stores. The specs are:

Sonalert model SNP428
Volts 4 — 28 VDC
Amps .003 — .016
Manufactured by P. R. Mallory & Co., Inc.

All other materials are readily available at radio supply/hobby stores.
The procedure for equipping the COMPUCOLOR II with sound is as follows:

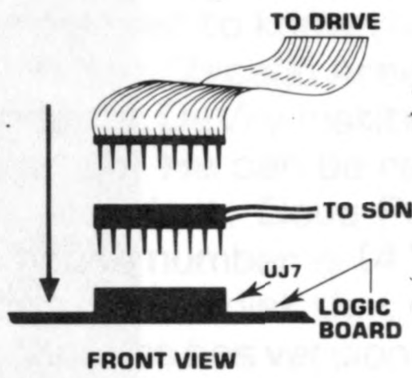
1. Add the IN914 diode to the Sonalert by soldering the negative side of the diode to the plus side of the Sonalert; and the positive side of the diode to the negative side of the Sonalert.
2. Attach the two 12 inch wires. A red wire for the +5VDC (+) side of the Sonalert, and a black wire for the minus (−) side of the Sonalert. Lightly twist the wires together. Bare the free ends of the wires and tin them with solder.
3. Attach the red wire to a +5VDC location on the logic board. One good place is Pin 8 of the J7 (internal disk drive connector). The black wire can be connected to Pin 6 of J7.
4. As J7 is a socket that the internal disk drive plugs into, the best method of connecting the Sonalert is with another socket. Note the drawing below:

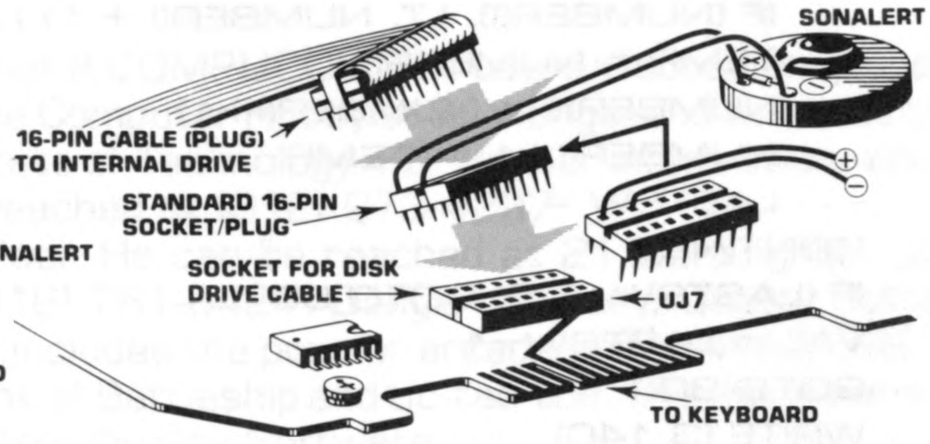
VIEW OF COMPUCOLOR II LOGIC BOARD

Connect (+ Plus) side of SONALERT to pin 8 of UJ7.

Connect (− Minus) side of SONALERT to pin 6 of UJ7.

Note: For COMPUCOLOR II owners whose disk drive is mounted externally, no cable is used inside the machine to attach the disk drive. Therefore, the UJ7 socket is unused, and only the 16-pin plug is needed to attach the SONALERT.





5. Mount the Sonalert onto the inside of the COMPUCOLOR II cabinet. You can do this by gluing the speaker in place, or by using a bracket. The purists (and the daring!) can cut a hole into the COMPUCOLOR II cabinet and mount the Sonalert through the hole.

KEEPING IT SIMPLE

introduction to fortran

Largely because of its easy-to-use English syntax and its general purpose nature, BASIC has become the standard language for personal computing. Still, BASIC does have its shortcomings as far as speed and flexibility are concerned, and some applications are much more conveniently written in a different computer tongue. Assembly language has been available on the COMPUCOLOR II since its inception, and many of our most popular programs are written in it. In order to expand the capabilities of your COMPUCOLOR II, we now offer Microsoft FORTRAN as a $75 option.

FORTRAN was first developed in 1954, which is practically pre-Cambrian on the computer time scale. But FORTRAN was carefully designed and has continued to grow and develop over the last 20 years such that its popularity remains very high. A high percentage of serious computer installations use FORTRAN in one form or another.

FORTRAN has some very specific advantages on the COMPUCOLOR II. FORTRAN is fast — almost as fast as assembly language. FORTRAN allows you to generate the fast-moving graphics necessary for real-time video-game applications. But FORTRAN is relatively easy to learn — almost as easy as BASIC. In fact, you can even write a program in BASIC and then simply translate it into FORTRAN, since FORTRAN and BASIC have many similarities. But FORTRAN is a higher level language than BASIC because it is compiled, not interpreted. This means that FORTRAN, when read by the COMPUCOLOR II, actually generates assembly language code, whereas BASIC does not.

FORTRAN also allows you to have more formatting control over hard copy, meaning that your output can be tailored to precise specifications. FORTRAN has much to offer for both experienced and inexperienced users. In order to make FORTRAN available to as many users as possible, we have priced it very well, far below a usual single-copy price.

Creating a working FORTRAN program requires three steps. First, the program is **written** using an editor such as our screen editor. Then the program is **compiled** — translated into machine language in a relocatable format. Thirdly, the program is **linked**. The Linking Loader uses a library file to look up all of the routines that will be necessary to run the program. From these routines the linker produces a runable .PRG program.

The program below introduces you to FORTRAN to let you get a taste of this popular computer language. Note that it is somewhat different from BASIC in appearance. In FORTRAN, line numbers are not necessary on every line, since statements are always processed in sequential order. Note also that all logical comparisons in FORTRAN are called with a two-character name surrounded by points.

The FORTRAN "DO" loop is similar to a "FOR NEXT" loop, except that it has some different rules about what can be contained in the loop. This program does an exchange sort on integers. Even those of you who do not plan on

KEEPING IT SIMPLE . . . Cont.

KEEPING IT SIMPLE . . . Cont.

investing in this second language for your CCII should find exposure to this widely-used computing language worthwhile.

```
C  INTEGER EXCHANGE SORT (20<N<0)
       PROGRAM SORT
       INTEGER VAL,NUMBER (20),TEMP
10     WRITE(3, 100)
       READ(1, 110) N
       IF (N.GT.20) GOTO 10
       DO 20 I=1,N
          WRITE(3, 120) I
          READ(1, 130) NUMBER (I)
20     CONTINUE
       VAL = N-1
30     LASTSW = 0
       DO 40 I=1,VAL
          IF (NUMBER(I). LT. NUMBER(I + 1) ) GOTO 40
          TEMP = NUMBER(I)
          NUMBER(I) = NUMBER(I+1)
          NUMBER(I+1) = TEMP
          LASTSW = I
40     CONTINUE
       IF (LASTSW.LT.2) GOTO 50
       VAL = LASTSW-1
       GOTO 30
50     WRITE (3, 140)
       DO 60 I=1,N
          WRITE (3, 150) NUMBER(I)
60     CONTINUE
100    FORMAT('+HOW MANY VALUES TO SORT?')
110    FORMAT(I3)
120    FORMAT('+#',I3,'>')
130    FORMAT(I6)
140    FORMAT(/,' THE SORTED LIST:',/)
150    FORMAT(1X,I6)
       STOP
       END
```

BOOK REVIEWS

We've decided that a place for book reviews is definitely needed in Colorcue, since most of you report that books have been a prime source of your computer knowledge. In the past months, several of you have written in to let us know of your experiences with various volumes, and we appreciate your taking the time to keep us informed. Now we'd like to return the efforts by supplying, in every issue, reviews of two computer books which are generally available. We've chosen a review format which will make deciphering our comments easy. We will answer these questions about each book:

1. At readers of what programming level is the book aimed?
2. How adaptable are the book's programs and theory to the COMPUCOLOR II?
3. What is the overall usefulness of the book?

PROBLEMS FOR COMPUTER SOLUTION — Donald Spencer (Hayden Publishing) Paperback, 125 pages, a few diagrams, mostly text.

1. This book assumes that the reader has some understanding of a computer language. It offers problems of varying degrees of difficulty and of various types. The book is divided into subgroups according to subject matter. The book can be used by either a teacher or student as an instructional aid, and is also useful for the hobbyist. There are no programming instructions in any language, nor does the book assume that the problems will be worked in a certain language. No solutions are given to the problems in the book, answers must be obtained through individual perseverance.

2. The book is adaptable to any computer, including the COMPUCOLOR II. All the programs can be written and solved on the COMPUCOLOR II without any difficulty since the book does not demand any specific language.

3. **PROBLEMS FOR COMPUTER SOLUTION** is a great aid to someone learning a computer language, especially BASIC. The author seems to subscribe to the maxim that experience is the best teacher, and the book offers the user a chance to get lots of experience. This book is potentially very useful for teaching BASIC if used with another book which gives specifics about the BASIC language. Since the book has a generic approach as far as

languages are concerned, and does not specify any one language, the programmer can use this text over and over again as he sets about becoming a computer polyglot. This book is recommended to anyone in the process of learning a computer language.

HOME COMPUTERS CAN MAKE YOU RICH — Joe Weisbecker (Hayden Publishing) Paperback, 119 pages, no listings, a few pictures.

1. This book talks to people who are interested in making money from microcomputers. It is relatively jargon-free and written so that anyone who has a basic understanding of what a computer is can start profiting from his knowledge. The book explains how to profit either from one's own efforts, or through hiring someone else. The book does not require extensive background in computers, and is directed at the average person who has some interest in the growing computer market.

2. The book contains a great deal of information about how to offer a computer service or program. This information is readily adaptable to the COMPUCOLOR II.

3. The book is informative and offers interesting advice to anyone who desires to profit from personal computers. It gives some new ideas on how to sell in this specialized market. For the person with a serious interest in selling software, the book is certainly recommended.

USERS NEWS

clubs

Those of you in the Chicago area will be pleased to know that a COMPUCOLOR II users/discussion group has been formed. It will be a sub-section of CACHE, the Chicago Area Computer Hobbyist Exchange, and will meet at the regular CACHE meetings — every third Sunday at DeVry Institute of Technology. For further information, contact Bill Cody, who is organizing the discussion group. He can be reached at (312) 973-4237.

The Canadian Users Group has a new president, Doug Peel. He can be reached at 21 Dersingham Crest, Thornhill, Ontario, CANADA L3T4P5. The phone number is (416) 751-8421. Doug's company, Quality Software Associates, has just come out with an entertainment disk that includes the popular arcade game INVADERS, with sound that doesn't require Soundware. The disk also has versions of Battleship and tic-tac-toe. The disk is marketed through Compucolor dealers or it can be purchased directly from Quality Software.

correspondents

Any users in the Anchorage area can contact:

Arthur Lawton, Jr.
SRA Box 1721A
Anchorage, AK 99507

He'd like to get some dialogue going with fellow COMPUCOLOR II owners. And in Toledo, Ohio,

Doug Loomis
5850 Yarmouth
Toledo, OH 43623

would like to start a group of users in his area. If you're in northern Ohio, drop him a line.

creativity abounds

One of the COMPUCOLOR II owners in the Huntsville, Alabama area has an artistic bent. His name is Tony O'Neil and he manufactures solid bronze belt buckles customized to any design specifications. His most recent effort is the ISC logo. He has cast it in a 2x3 chunk of bronze which fits belts up to an inch and a half wide. If you're interested in sporting this designer label, priced at $8.00, order from

The Bronze Bear
P.O. Box 2251
Huntsville, AL 35804

Postage is $1 domestic, $2 foreign.

history library

Graphic-History, a company in Atlanta, Georgia, has developed a fascinating package for the COMPUCOLOR II which uses color and graphics to describe World War II's Normandy Invasion. The program is a good one — easy to use and very informative. The end-user price is $39.95 (plus $2.00 shipping) and the package, which includes a Sof-Disk and documentation, can be obtained from:

Graphic-History
35 Executive Park Dr., NE
Atlanta, GA 30329

For more information, call Mark Whitworth or Carlton Joyce at (404) 321-7910. Graphic History is working on a series of historical programs that will become available over the next year.

Intelligent Systems Corp.®

225 Technology Park/Atlanta
Norcross, Georgia 30092
Telephone (404) 449-5961
TWX 810-766-1581